A RAND NOTE

STRATEGIES OF COOPERATION IN DISTRIBUTED PROBLEM SOLVING

Stephanie Cammarata, David McArthur,
Randall Steeb

October 1983

N-2031-ARPA

Prepared for The Defense Advanced Research Projects Agency

35th Year

Rand
SANTA MONICA, CA. 90406

The research described in this report was sponsored by the Defense Advanced Research Projects Agency under ARPA Order No. 3460, Contract No. MDA903-82-C-0061, Information Processing Technology Office.

Published by The Rand Corporation

A RAND NOTE

STRATEGIES OF COOPERATION IN DISTRIBUTED PROBLEM SOLVING

Stephanie Cammarata, David McArthur, Randall Steeb

October 1983

N-2031-ARPA

Prepared for The Defense Advanced Research Projects Agency

35th Year

Rand
SANTA MONICA, CA. 90406

PREFACE

This Note summarizes the results of an experimental investigation of techniques for distributed problem solving, conducted for the Information Processing Technology Office, Defense Advanced Research Projects Agency (DARPA), under Contract No. MDA903-82-C-0061. The work has focused on development of organizational structures for cooperative planning in complex, spatially distributed systems, using air traffic control as an illustrative context.

SUMMARY

Distributed artificial intelligence is concerned with problem solving that is done by groups of agents. This Note describes strategies of cooperation that groups require to solve shared tasks effectively. We discuss such strategies first in a domain-independent fashion, and then in the context of a specific group problem-solving application: collision avoidance in air traffic control. We begin by contrasting the methodologies, difficulties, and opportunities of distributed and centralized problem solving. From this analysis, we infer a set of requirements on the information-gathering and organizational policies of group problem-solving agents. We then discuss a set of distributed problem solvers that we have developed in the domain of air traffic control and describe some experimental findings with the cooperative strategies used. In particular, we note large task-dependent differences in processing times, communication loads, and system errors between the several cooperative strategies.

ACKNOWLEDGMENTS

We appreciate the assistance and advice of William Giarla, Sanjai Narain, and Philip Klahr.

CONTENTS

FIGURES

I. INTRODUCTION

Distributed artificial intelligence is concerned with problem solving that is done by groups of agents. Through systems like STRIPS [5], ABSTRIPS [10], BUILD [4], and NOAH [11], we have gained some understanding of how a single agent can solve problems. Unfortunately, the representations of knowledge [6,1] and planning expertise [9] required of agents in distributed or group problem solving appear to be quite different from those required of single-agent problem solvers.

In this Note we focus on one particularly important but little-understood topic: the kinds of *strategies of cooperation* that groups require to solve shared tasks effectively. We will discuss such strategies first in a domain-independent fashion and then in the context of a specific group problem-solving application. We begin with a general discussion of the difficulties and opportunities facing distributed problem solvers in many domains, as contrasted with those of single-agent problem solvers. From this analysis, we infer a set of requirements on the information-gathering and organizational policies of group problem-solving agents. We then discuss a set of distributed problem solvers that we have developed in the domain of air traffic control. We concentrate on the particular cooperative strategies used, how they are implemented, and how they successfully overcome some of the obstacles that make it difficult to coordinate groups of agents.

II. DISTRIBUTED AND SINGLE-AGENT PROBLEM SOLVING

To understand how agents solving problems in a distributed fashion differ from single-agent problem solvers, we begin by studying the characteristics that distinguish distributed problems from single-agent problems.

CHARACTERIZATION OF DISTRIBUTED PROBLEM SOLVING

There are several general characteristics of distributed problem-solving situations that are particularly important for our purposes:

- Most situations consist of a collection of agents, each with various *skills*, including sensing, communication (often over limited-bandwidth channels), planning, and acting.
- The group as a whole has a set of assigned *tasks*. As in single-agent problem-solving situations, these tasks may need to be decomposed into subtasks, not all of which may be logically independent. The group must somehow assign subtasks to appropriate agents.
- Each agent typically has only *limited knowledge*. An agent may be subject to several kinds of limitations: limited knowledge of the environment (e.g., because of restricted sensing horizons), limited knowledge of the tasks of the group, or limited knowledge of the intentions of other agents.
- There are often limited shared *resources* that each agent can apply to tasks. For example, if the agents are in a blocks-world environment, the shared resources are the blocks of which their constructions must be made.
- Agents typically have differing *appropriateness* for a given task. The appropriateness of a particular agent for a task is a function of how well the agent's skills match the expertise required to do the task, the extent to which his limited knowledge is adequate for the task, and his current processing resources.

DIFFICULTIES IN DISTRIBUTED PROBLEM SOLVING

There are several difficulties that can arise when solving problems in a distributed fashion that are not significant in most single-agent problem-solving situations. First, in single-agent problem solving, the agent is typically *given* his task as part of the problem definition [10,4], whereas in distributed situations, the *assignment* of tasks to the agents is part of the group problem-solving activity. This assignment can be challenging. Many mappings of tasks to agents are possible, but because agents typically have differing available expertise for a given task, only a few agents will be acceptable for each task. Thus in many distributed problems, it is crucial for agents to adopt the right *role*. It would not be reasonable to assign the role of inventing a new chip to a lawyer, or the role of writing the patent to an engineer. In addition to ensuring that each task is assigned to an acceptable agent, the group has to ensure *task coverage*. Specifically, this means that all tasks should be assigned some agent (complete role assignment) and that extra or redundant agents should not be assigned tasks (consistent role assignment). For example, in air traffic control, if the task is to solve a possible spatial conflict, it is critical to ensure that only one aircraft detours; if two or more adopt that role, they may possibly create a new collision situation.

Compounding the difficulty of finding an optimal task assignment is the limited knowledge of the agents. In most single-agent problem solvers, the agent has a complete world model, which usually remains complete because (1) all changes in the environment are made by the agent and thus he can always update his world model, and (2) a single agent does not have to worry about unknown intentions of other agents. The incomplete or incorrect world models of distributed agents may degrade the accuracy of assignment of tasks, either because agents who know the tasks may not know the agents with the most appropriate available expertise, or conversely, because the agents with the best expertise may not know about appropriate tasks for them. Similarly, the incomplete knowledge of agents may prevent consistent and complete role assignment because there may be no one agent who has a global knowledge of all the roles or subtasks that need to be assigned. In a single-agent problem-solving situation, this issue does not arise. The agent knows how he has decomposed a task into subtasks, and he knows exactly which subtasks he has to do--all of them.

Once tasks or roles have been assigned, distributed problem solvers face severe difficulties in *coordinating task execution*. Like single-agent tasks or subtasks, group tasks may not be independent. Temporal or logical dependencies may exist. For example, if the group problem is to build a new chip, the designer's role must be completed prior to the initiation of the manufacturer's. In addition, tasks that are not logically connected may interact through shared resources. For example, if two blocks-world agents are each to build towers, one agent's plan will negatively interact with the plan of another if both intend to use the same block [2]. The interaction is negative because the first agent is satisfying his task, but at the cost of preventing the second agent from satisfying his. In contrast, the plans might interact positively if one agent's plan entailed using (hence picking up) a block that currently lies on top of the block another agent intends to use. The interaction is positive in the sense that the first agent is not only satisfying his task, he is also helping the second agent satisfy his.

While single-agent problem solvers have difficulties in handling non-independent tasks or subgoals [14], these difficulties multiply for distributed problem solvers. Again, limited knowledge is the reason. If two agents have only local knowledge--if they know only the local environment and only their own tasks and intentions--they will not be able to prevent negative interactions between goals or roles. If the chip designer does not know about the chip manufacturer, there is no basis for coordinating their subtasks; if one blocks-world agent doesn't know the intentions of another, there is no basis for ensuring that their projected uses of resources will not conflict. Similarly, without some knowledge of others' tasks and intentions, positive interactions, the essence of effective group problem solving, cannot be encouraged.

In summary, the main challenge in distributed problem solving is that the solutions a distributed agent produces must not only be locally acceptable, achieving the assigned tasks, but they must also be interfaced correctly with the actions of other agents solving dependent tasks. The solutions must not only be reasonable with respect to the local task, they must be *globally coherent*, and this global coherence must be achieved by *local computation alone*. Global coherence is less difficult to achieve for a single-agent problem solver, simply because his computation and knowledge are themselves as global as the task requires.

III. STRATEGIES FOR COOPERATION

We have come to believe that there are no general algorithms to dictate optimum cooperation. Methods that yield good distributed performance under one set of conditions fail under others. Instead, cooperative expertise seems to take the form of a broad range of heuristic rules. We have classified these cooperative heuristic strategies under two headings: *organizational policies* and *information-distribution policies*.

ORGANIZATIONAL POLICIES

Organizational policies dictate how a larger task should be decomposed into smaller (sub)tasks which can be assigned to individual agents. Typically, a given organizational policy assigns specific roles to each of the agents in a group. Such a policy is useful if for some tasks the resulting division of labor enables agents to work independently. For example, the corporate hierarchy is an organizational policy that is particularly effective if the corporate task can be decomposed in such a way that an agent at one level can work independently of others at that level, reporting results only to his immediate superior, who takes care of any necessary interfacing.

Organizational policies not only define a task decomposition, but they also prescribe communication paths among agents. They turn a random collection of agents into a network that is fixed, at least for a given task. In the corporate hierarchy, again, the arcs between agents usually indicate which pairs are permitted to talk to one another, and, in addition, they determine the nature of the messages that are allowed. Such communication restrictions are beneficial if they encourage only those agents who should communicate to do so--in particular, agents who have dependent tasks or who may share resources. In general, organizational policies strongly direct and constrain the behavior of distributed agents. If those constraints are appropriate to the task at hand, then the organization is effective; otherwise, its performance may be suboptimal.

In our distributed problem-solving systems and others [12,3], groups begin by establishing an organizational policy. To do so, the agents must know not only which policy is appropriate to the current circumstances, but also the *techniques by which a group can implement the chosen policy in a distributed fashion*. Briefly, any distributed method of implementing an organizational policy must answer a variety of questions, including:

- When does organization structuring take place?
- How is the assignment of roles specified by the policy made to agents? In other words, how is the agent who is most appropriate for a given task found?
- Are agents externally directed or data-directed? [8] That is, does an agent arrive at his roles by being told them, or is information relayed, allowing him to make the assignment of roles himself?
- When an agent is requested by another agent to conform to a role, or to take on another subtask, does the first agent have the right to negotiate? How does an agent weigh the value of competing tasks?

Smith [12] has proposed the contract net as a formalism for implementing organizational policies in a distributed fashion. In Section V, we discuss how some organizational policies were imposed in a distributed air traffic control situation.

INFORMATION-DISTRIBUTION POLICIES

An information-distribution policy addresses the nature of communication between cooperating agents. Decisions about how agents communicate with each other are, first of all, constrained by the choice of organizational policy, since that policy decides the network of permissible communicators. However, within these constraints, a great number of lower-level decisions must be made about how and when communications should occur:

- *Broadcast or selective communication.* Are agents discriminating about who they talk to? If so, what criteria are used to select recipients?
- *Unsolicited or on-demand communication.* Assuming an agent knows who he wants to communicate with, does he do so only if information is requested, or does he infer the informational needs of other agents and transmit data accordingly?
- *Acknowledged or unacknowledged communication.* Does an agent indicate that that he has received information?
- *Single-transmission or repeated-transmission communication.* Is a piece of information sent only once, or can it be repeated? How frequently? Lesser et al. [7] refer to a repeated-transmission policy as murmuring.

Poor decisions at this level result, at best, in the highly inefficient use of limited-bandwidth channels. At worst, such choices endanger global coherence by preventing agents whose tasks may interact from talking to one another. The goal of information-distribution policies is to minimize these possibilities. As with organizational policies, the utility of communication policies depends on current conditions. These include the bandwidth of the communication channel, the reliability of the channel, the load of the channel, the maximum acceptable information turnaround time, and the relative cost (and time) of computation versus communication.

IV. DISTRIBUTED PROBLEM SOLVING IN AIR TRAFFIC CONTROL

Problem solving in air traffic control (ATC) may be distributed in several ways. In an earlier report [13], we discuss a variety of architectures of distribution. Currently, we have implemented only *object-centered* systems, where one agent is associated with each aircraft. In our ATC task, aircraft enter a rectangular (14 x 23 mile) airspace at any time, either at one of 10 infixes (entry points) on the borders of the airspace, or from one of two airports. The main goal of the agent is to have the aircraft he is associated with traverse the airspace to an assigned destination--either a boundary outfix or an airport. Each aircraft has only a limited sensory horizon, hence its knowledge of the world is never complete and it must continually gather information as it moves through the airspace. Information may be accumulated either by sensing or by communication. Agents are allowed to communicate over a limited-bandwidth channel to other aircraft for purposes of exchanging information and instructions.

Distributed ATC is a group problem not only because agents may help one another gather information, but also because the goals of one agent may interact with those of another. Goal interactions come in the form of shared conflicts. A conflict between two or more agents arises when, according to their current plans, the two will violate minimum separation requirements at some point in the future. When shared conflicts arise, agents must negotiate to solve them. In a crowded airspace, such goal conflicts can get particularly complex and may involve several aircraft, thus necessitating a high degree of group cooperation.

In terms of the vocabulary developed in Section II, the detection and resolution of conflicts are the main distributed problem-solving *tasks*. These tasks may be decomposed into several *subtasks* or distinct *roles*. Agents may gather information about a shared conflict, evaluate or interpret the information, develop a plan to avoid a projected conflict, or execute such a plan. Agents may be more or less *appropriate* for such roles, depending on their current processing load (are they currently involved in helping resolve other conflicts?), their state of knowledge (do they know a lot about the intentions of other agents in the conflict?), and their spatial constraints (can they see many nearby aircraft and do they have much excess fuel?).

The issue of *optimal task assignment* arises because a group of aircraft may fail to assign the most appropriate agent to each role in a conflict task if some of the aircraft do not know about a shared conflict. In addition, care must be taken to assign a complete and consistent set of roles. Some role inconsistencies can be fatal. For example, two agents would be adopting inconsistent roles if one decided to move left to avoid a head-on collision with the second, while the second decided to move right. Severe *task coordination problems* may also arise in distributed ATC. The action of moving to avoid one conflict may create or worsen other conflicts (negative task interactions) or lessen other conflicts (positive task interactions). Both forms of interaction are caused by the fact that while agents may be dealing with different conflict tasks, they are nevertheless exploiting shared limited spatial resources.

V. FOUR DISTRIBUTED PROBLEM SOLVERS FOR AIR TRAFFIC CONTROL

We now outline the cooperative strategies embedded in four distinct ATC systems. All four systems are implemented in our framework for constructing distributed agents [9]. This in turn is implemented in INTERLISP-D, running on Xerox 1100 computers.

INFORMATION-DISTRIBUTION POLICIES IN ATC

The information-distribution policy common to all four systems prescribes that information should be sent to other aircraft selectively (no broadcasting), without waiting for a request, without expecting an acknowledgment, and without repeating the information a second time. These choices are reasonable, since we assume in all systems that communication is error-free. When we add noise to the communication channel, we envision adopting a policy that injects some needed redundancy or safety into communication, for example, a policy that includes murmuring [7]. We also assume a constant effective communication bandwidth for all four systems. Each aircraft is allowed to send a maximum of 5 messages per 15 seconds of time.

ORGANIZATIONAL POLICIES IN ATC

The organizational policy embedded in three of the four systems may be characterized as *task centralization*; the fourth system adheres to a policy of *task sharing*. Under *task centralization*, the agents involved in any given conflict task will choose one of their number to play most of the roles. In particular, one agent will perform the evaluation role (do all the evaluation of the potential conflicts between aircraft), the plan-fixing role (attempt to devise a plan-fix to dissolve the entire conflict), and the actor role (act on the new plan). The selected agent is required to modify only his plan to resolve the conflict; thus the remaining agents perform no planning or actions. Instead, having agreed on the choice of a replanner, they adopt passive information-gathering roles, merely sending their intentions (plan) to the selected agent. However, if the selected agent is unable to resolve the entire conflict, he requests another agent to replan. This process continues until all conflicts are resolved or a solution cannot be found. The policy of task centralization, whatever its shortcomings, is worth considering because it enjoys many of the advantages of the centralized, single-agent problem solving that it is meant to mimic. Specifically, by centralizing most task roles in a single agent, the group has to worry less about negative task interactions such as the threat of two aircraft acting in an inconsistent fashion, noted above.

Although three of our four systems embed a task centralization policy, they differ in how they measure and choose the agent who is the most appropriate for the several centralized roles.

Selection by Shared Convention. Here, each aircraft uses only directly sensed information about the other aircraft (position, heading, and speed) to decide who should plan and who should transmit his current route. The aircraft silently use a common set of conventions for this decision, minimizing communications. Figure 1 shows a prototypical sequence of tasks and communications between two aircraft under this policy.

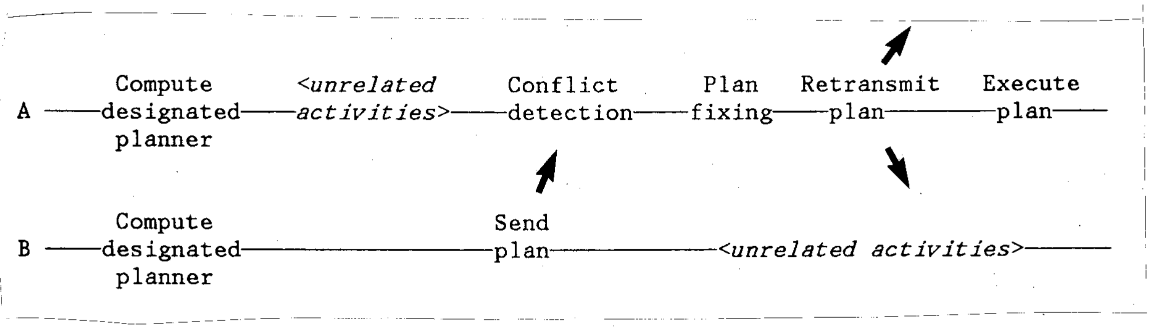

Fig. 1 -- Prototypical task sequence under the shared convention policy. Time lines for tasks executed by aircraft A and B. Arrows indicate communications.

Because of the limited criteria used, the aircraft selected as the replanner is not likely to be the most appropriate. This version mainly serves as a benchmark against which to judge the utility of more intelligent methods of selection which are also more costly in terms of computation and communication.

Selection of the Least Spatially Constrained Agent. Here, each aircraft in a potential conflict transmits its *constraint factor* to the other aircraft. The constraint factor is an aggregation of such considerations as the number of other nearby aircraft, fuel remaining, distance from destination, and message load. Figure 2 shows the standard sequence of tasks and communications under this policy.

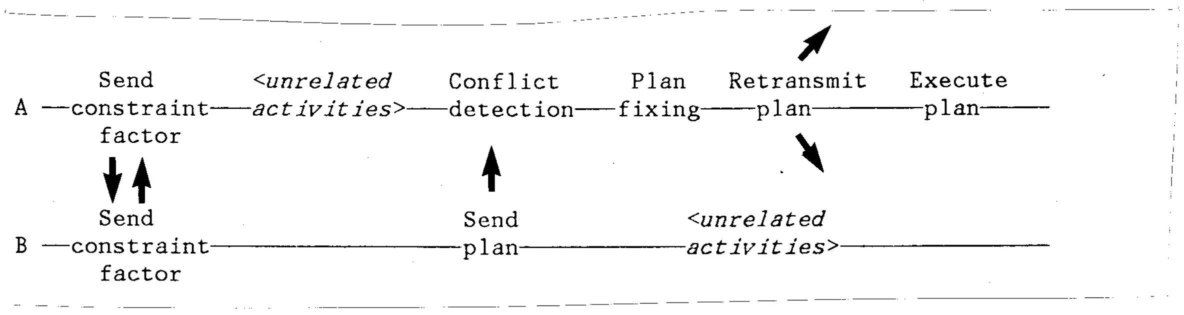

Fig. 2 -- Prototypical task sequence under the least spatially constrained policy.

This method of selection maintains that the most appropriate agent is the one with the most degrees of freedom for modifying his plan. It is a more complex process than the *shared convention* and should result in more effective replanner choices, although at some additional cost in initial communications.

Selection of the Most Knowledgeable, Least Committed Agent. As above, aircraft share *constraint factors*, but here they are computed differently. This method of selection maintains that the best agent to replan is the one who knows the most about other agents' intentions, because, in replanning, a well-informed agent can explicitly take account of possible interactions between his intentions and those of other agents. More globally coherent plan-fixes should therefore result. In addition, this method says that agents whose intentions are known by others should not replan. If such an agent does modify his plan, he will have violated the expectations of cooperating agents, making their knowledge incorrect and in turn making cooperation difficult. Thus, this policy implements a common adage of cooperation: Don't do the unexpected.

In spite of their simplicity, task centralization policies are often ineffective. Although the agent selected to perform the centralized roles may be the best *overall*, that agent is rarely the best for *each* of the centralized roles. For example, we still might want to assign the actor role to the agent in a conflict set who is least constrained in the sense defined above. However, that agent might not be the best in the set for fixing his plan--for making a modification to the plan and evaluating the implications of such a change. Presumably the best agent for this role is the (possibly distinct) member of the conflict set that knows most about the environment and intentions of aircraft near the one whose plan is to be fixed. This aircraft is in the best position to determine whether any changed plan is not only locally reasonable, solving the conflict, but also globally reasonable, not creating new conflicts with other aircraft.

Our *task-sharing* policy attempts to avoid such problems by evaluating agents' qualifications with respect to each of the roles associated with a conflict. While in centralized policies a single negotiation determines an overall replanner, in the task-sharing policy two rounds of negotiation are necessary, one to determine the plan-fixer and one to determine the actor. Figure 3 presents a prototypical sequence of tasks and communications showing how such a policy is implemented in a distributed fashion.

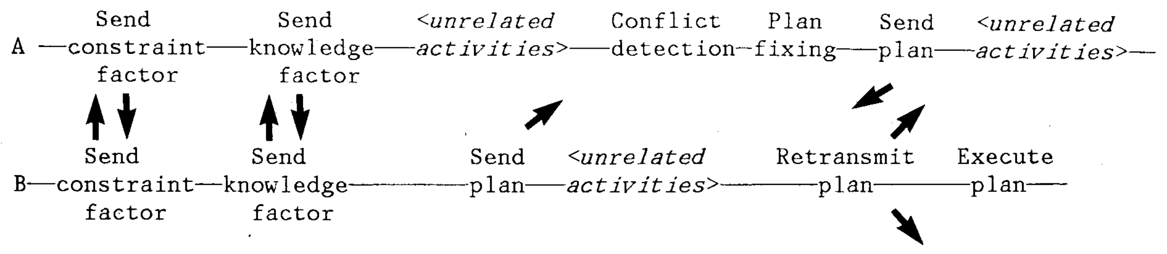

Fig. 3 -- Prototypical task sequence under the task-sharing policy.

Performance of groups working under a task-sharing policy is potentially superior to that of groups working under a policy of centralization, because in the former the group attempts to optimize on each role. However, in practice, this policy has several possible drawbacks. It is communication-intensive and may be inappropriate when communication channels are unreliable or costly. Moreover, it risks potential negative interactions, because several agents have to coordinate intimately to achieve a solution.

VI. EXPERIMENTAL STUDIES

DESCRIPTION

We conducted a series of rudimentary experimental studies on the four policies outlined above. We report here on results pertaining to only the three task-centralization policies, since our studies on the task-sharing policy were performed later and were limited in scope. The three task-centralization variants were tested on eight distributed scenarios. Each scenario stipulated (1) how many aircraft would enter the airspace in the session, (2) when and where they would enter, and (3) where they would exit. This control over the parameters of distributed problem-solving situations allowed us to isolate situation features that uncovered the strengths and weaknesses in performance of our policies. In particular, the scenarios varied considerably in task density, time stress, and task difficulty. The primary factor affecting these conditions was the number of aircraft in simultaneous conflict.

We examined three performance indices when comparing the systems: communication load, processing time, and task effectiveness. Task effectiveness was indicated by two distinct factors: separation errors (more important) and fuel usage (less important). A summary of the main results is given in Table 1.

DISCUSSION OF RESULTS

We found that the *shared convention* policy, relying on essentially arbitrary assignment of planning responsibility, performed well only in low-complexity, low-difficulty tasks. It minimized communications and response times compared with the other policies, but it quickly foundered in 3- and 4-body conflicts.

The *least constrained* policy performed best overall. It did particularly well on high-complexity, high-difficulty tasks. In such cases, the planning aircraft tended to be located at the edge of the fray, able to find more viable solutions than the aircraft in the interior. The policy was time- and communication-intensive, however, largely because of the high number of messages needed to cooperatively

Table 1

PERFORMANCE MEASURES OF THREE ORGANIZATIONAL POLICIES
(statistics averaged across 8 scenarios)

Item	Shared Convention	Least Constrained	Most Knowledgeable
Communication load [a]	10.9	28.6	28.2
Processing time [b]	1265	1726	1651
Separation errors [c]	4.3	1.4	2.3
Fuel usage [d]	96	108	101

[a] *Communication load* = mean messages sent per aircraft while flying from infix to outfix.

[b] *Processing time* = mean Xerox 1100 cpu seconds per aircraft while flying.

[c] *Separation errors* = mean number of near misses or collisions for all aircraft in a scenario.

[d] *Fuel usage* = mean number of fuel units used for all aircraft.

determine the replanner and to maintain consistency after replanning. In any of the three systems, when a replanner is successful he must send *data retransmission messages* to all aircraft to which he had previously sent his intentions. The number of data retransmissions was especially high under the *least constrained* policy.

The *most knowledgeable* policy was intermediate in performance. It performed best in tasks of low complexity and high difficulty, that is, tasks with primarily 2- and 3-body interactions but few potential solutions. In complex multi-aircraft situations, if the wrong aircraft was chosen for planning, the result was often catastrophic, because the aircraft that then received replan requests tended to have little knowledge of the routes of other aircraft. By design of the policy, this knowledge was typically concentrated in the initially selected planner.

When successful, the *most knowledgeable* policy's performance was in some ways better than that of the *least constrained* policy. In particular, when an agent found a solution to a local conflict task under the *most knowledgeable* policy, his solution was likely to be more globally coherent than solutions found under other policies, since the replanning agent was selected partially because of his wide knowledge of the plans of the other aircraft. This knowledge allowed him to more effectively replan without incurring new conflicts. In addition, a successful replanning agent under a *most knowledgeable* policy generally needed to issue fewer data retransmission messages than under the other policies, since he was selected to replan partially because his intentions were known to fewer others (i.e., he was the least committed agent). We had initially anticipated that minimizing data retransmissions would be very important for guaranteeing globally coherent performance. We envisioned situations where one retransmission would cause the receiving agent to reevaluate, possibly finding new conflicts, causing more replanning, further data retransmissions, and so on in a vicious propagation of changes. This did not happen as often as we had expected under the the *least constrained* policy, although a few instances were observed.

Another erroneous expectation was that there would be a wide variation in processing times among the aircraft under the *most knowledgeable* policy. This policy should tend to bias replanning in favor of a few agents. If an agent is the replanner once, he gains new knowledge of others' plans, making him an even better choice as re-planner for later conflict tasks. We anticipated that this concentration would skew the processing times compared to a more uniform distribution of responsibilities under the other policies. This would have been a disadvantage in a truly distributed system, as some agents would be quiescent much of the time. The expected variation in times did not evidence itself, however, except in the relatively easy scenarios.

While limited in scope, the data collected from our fourth policy, *task sharing*, indicated some interesting trends. This policy, a composite of the best of the *least constrained* and *most knowledgeable* policies, had the advantage of choosing one agent to act, and another with more knowledge of the situation to compute the first agent's plan. We found that this policy was often effective in situations where subtasks are easily separable and an explicit selection of the agent with the best available expertise could be made for each subtask individually, rather than for the conflict task as a whole. We plan to study this policy further in our future simulations.

VII. CONCLUSIONS

Distributed problem solving is an enigma. Potentially, a group of agents working together should be able to solve problems more effectively than the same agents working individually. In practice, however, groups often work ineffectively and their joint productivity is less than the sum of the productivities expected of each member. Our aim is to discover the elusive cooperative strategies that enable groups to reach optimum productivity. On the theoretical side, we are developing concepts that allow us to simply and formally describe various cooperative strategies. On the empirical side, we are testing such strategies by imposing them on groups and observing the resulting group performance. Both phases are necessary. The theoretical investigations are valuable because most presently available problem-solving models describe only the information processing of single-agent problem solvers, not distributed problem solvers. The empirical work is necessary because many of the behavioral properties of complex cooperative strategies are not apparent until the strategies are employed in real or simulated settings.

REFERENCES

1. Appelt, D. E., Planning Natural-language utterances to satisfy multiple goals, Technical Note 259, SRI International, 1982.

2. Davis, R., A model for planning in a multi-agent environment: steps toward principles for teamwork, Working paper, MIT Artificial Intelligence Laboratory, 1981.

3. Davis, R., and R. G. Smith, Negotiation as a metaphor for distributed problem solving, Memo 624, MIT Artificial Intelligence Laboratory, 1981.

4. Fahlman, S., A planning system for robot construction tasks, *Artificial Intelligence*, 5, (1), 1974, 1-49.

5. Fikes, R. E., and N. J. Nilsson, Strips: A new approach to the application of theorem proving to problem solving, *Artificial Intelligence*, 2, (2), 1971, 189-208.

6. Konolige, K., A first order formalization of knowledge and action for a multi-agent planning system, *Machine Intelligence 10*, 1981.

7. Lesser, V. R., S. Reed, and J. Pavlin, Quantifying and simulating the behavior of knowledge-based interpretation systems, *Proceedings of the First Annual National Conference on Artificial Intelligence*, Stanford University, 1980, 111-115.

8. Lesser, V., A high-level simulation testbed for cooperative problem solving, COINS Technical Report 81-16, University of Massachusetts, Amherst, 1981.

9. McArthur, D., R. Steeb, and S. Cammarata, A framework for distributed problem solving, *Proceedings of the National Conference on Artificial Intelligence*, Pittsburg, 1982, 181-184.

10. Sacerdoti, E., Planning in a hierarchy of abstraction spaces, *Artificial Intelligence*, 5, (2), 1974, 115-135.

11. Sacerdoti, E., A structure for plans and behavior, New York: Elsevier North-Holland, 1977.

12. Smith, R. G., A framework for problem solving in a distributed processing environment, STAN-CS-78-700, Stanford University, 1978.

13. Steeb, R., S. Cammarata, F. A. Hayes-Roth, P. W. Thorndyke, and R. B. Wesson, Distributed intelligence for air fleet control, R-2728-ARPA, The Rand Corporation, 1981.

14. Sussman, G., A computational model of skill acquisition, New York: American Elsevier, 1975.

RAND/N-2031-ARPA